THE USE OF NULL VALUES IN A RELATIONAL DATABASE TO REPRESENT INCOMPLETE AND INAPPLICABLE INFORMATION

by

MARIA MARSHALL WILSON

B. Ed., Washburn University, 1973

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1985

Approved by:

Major Professor

Contents

Table of Figures

Many thanks to Beth for her guidance
and constant optimism.

Chapter 1 : Introduction

"Much ado about nothing"

-Shakespeare

An increasing concern in relational database theory is that of retaining representational consistency while allowing additional semantic representation. A developed relational schema will not accept information that does not completely adhere to the preconceived structure. Therefore, incomplete sets of information may not be stored, and subsequently are not available for processing.

One proposed method for dealing with this problem is the extension of the relational model with the inclusion of nulls as valid storable and processable values. A null value is defined as a value which is incompletely specified or in some manner unknown or inconsistent. A report released by ANSI/X3/SPARC lists fourteen manifestations of nulls (see appendix a). This list can be generally categorized as three distinct types of nulls: unknown values, incomplete or inconsistent values, and values resulting from processing one of the previously mentioned types of null (the fourteenth manifestation).

Allowing representation of nulls is an important

issue in modeling possible real-world situations where unavailable information can prevent inclusion of related data in all processing. The artificial intelligence community has long recognized the need for more fully representing the semantic aspects of stored information. It is apparent that the constraints of representation imposed by normalization of relations severely limit the semantic flexibility of the relational database model.

The purpose of this study of nulls is to propose a method with which to represent incomplete or partial knowledge within the structure of the relational database model. The definition of a sound set of basic relational operations to apply to these relations allowing the presence of nulls is necessary to preserve the underlying integrity of the database. Allowing processing of relations containing partial or unknown information can also be of use in deriving inferential knowledge about such information.

Review of Literature

The scope of the research conducted includes several approaches to the solution of the problem of dealing with null values. Four principle categories are discussed: first order (predicate) logic, fuzzy

logic and non-monotonic reasoning, knowledge representation from the artificial intelligence viewpoint, and various relational algebras from relational database theory. The logicians are concerned with n-valued logics, partial order lattices, and proofs of representations of systems. Fuzzy logic approaches the problem from the aspect of inference, circumscription and non-monotonic reasoning, and domain sets. Artificial intelligence views deal with human inference, default reasoning, knowledge bases, and the constraints necessary to model the semantics of such systems. The relational algebras developed for null values include such varied representation methods as range domains, sets, and logical quantifiers. Database systems with no nulls, nulls representing unknown values, and marked nulls are discussed. These various approaches are outlined in the following section.

First-Order Predicate Logic

One of the views coming from the field of logic is that a three-valued logic must be defined to deal with an unknown truth value. A method introduced by Colmerauer [Col81], in a paper dealing with knowledge representation, transforms a complete three-valued logic into a corresponding two-valued logic. The notion of presupposition, which is necessary for natural

language processing, is supported in this view. Assertions are defined to contain two properties, and the resulting logic system is interpreted using the presupposition property.

Vassiliou [Vas79] abandons classical two-valued logic for what he considers a more appropriate n-valued logic (modal logic) to allow representations of nulls. He suggests redefinition of the interpretation of functional dependencies along with the requirements of satisfiability of inference rules. He also describes a many- valued logic approach with examples using denotational semantic interpretations to better understand the problems of dealing with null values [Vas80].

Jaegermann, in a two part discussion of information storage and retrieval systems with incomplete information [Jae78,Jae79], has developed a theoretical system incorporating descriptor algebras. These turn out to be pseudo-Boolean algebras forming a lattice of the describable sets of a given system.

Fuzzy Logic

The developing area of logic dealing with fuzziness of classification, known as fuzzy logic, was investigated. The focus of the papers from this area [Dav80,McC80,McD80] is on non-monotonic reasoning.

Circumscription, a form of this reasoning, is discussed as a method of allowing a formalized rule of conjecture to be used with the rules of inference of first order logic. Nonmonotonic logic deals with a system in which new axioms can invalidate previous theorems. Processes acting in the presence of incomplete information must possibly revise asumptions based on new observations.

Buckles [Buc82] presents a structure for representing inexact information in a relational database. He describes a fuzzy relational algebra as a special case of ordinary relational algebra. As equivalence is based on identity in normal relational theory, a weakening of dependence on this equivalence is necessary to allow manipulation of stored fuzzy relations.

Buell [Bue82] discusses fuzzy subset theory as it relates to information processing. His fundamental concern is that of the underlying lattice structure of a fuzzy subset.

Zadeh [Zad83] asserts that fuzzy reasoning allows fuzzy concepts such as "most", "many", "infrequently", and "about" to be modeled. Applicable mainly to rule-based expert systems design, fuzzy logic subsumes both predicate logic and probability theory.

Lipski [Lip79] states that fuzzy sets are not

applicable to this study, as the information is not inherently fuzzy, only incomplete (see Relational Algebras).

Artificial Intelligence Approaches

Artificial intelligence-based approaches are concerned with the semantically correct interpretation of information. Levesque [Lev81] presents a discussion on the interaction between an expert system and its knowledge base which contains incomplete information. He describes special expressive requirements necessary for the representation of incomplete information.

Gaines [Gai81] is concerned with the type of real-world information which can be represented only in softer terms than are currently in use. Data which is imprecise, dynamic, or redundant can not be incorporated into relational database systems unless a many-valued logic is used to determine the correct interpretation. He also attempts to model degrees of imprecision of data.

In a discussion of human inferential processing, Collins [Col75] suggests that added knowledge can increase uncertainty in some cases, thus invalidating the uniqueness assumption, and forcing the use of some default assumption. Lack of knowledge assumptions are also discussed. Functional inferences can aid

reasoning in the face of incomplete knowledge.

Reiter [Rei78,Rei81] presents a proof theoretic view of a relational database developed from a model theoretic representation. He examines the inclusion of nulls, integrity constraints, and conceptual modeling in terms of real-world semantics. He covers methods of using default reasoning to obtain semantically correct results from database query evaluations. To produce a proof of his model-theoretic database, Reiter must take a closed world view of the information his model represents. Every aspect of the model has a finite representability which enables his proof-theoretic model derivation.

Winograd [Win80] states that common sense reasoning differs from formal mathematical logic in the need to draw conclusions from partial information. Such reasoning is termed non-monotonic. His desire is to develop inferential systems which are efficient and are able to do plausible reasoning. He discusses such related issues as resource limitation and ordering of inferential processing.

Relational Algebras

Some background material from relational database theory was found to be of help. Codd's [Cod75]

original three-valued logic for null values is evidently one of the first attempts to define a representation to be used in database systems. Fagin [Fag82] finds that multivalued dependencies can be represented as simpler functional dependencies with a constraining join dependency, which would allow existing dependency theory to be more flexible in terms of representing nulls. Rissanen [Ris77] introduces the notion of attribute independence which aids in the definition of relations allowing null valued attributes.

Many of the interpretations of null values rely upon the underlying view taken of the stored information. Internal and external interpretations differ slightly in various models and so must be taken into account. In general, the external interpretation views the information in terms of the "real world," every possible information content the database could have. This interpretation is the "open world" view. The internal interpretation, on the other hand, must deal with information in the context of only what the database knows of the world--not reality itself, but only a restricted knowledge of reality. This is a "closed world" view, in which information not present is considered to be false information.

Much of the literature dealing with various attempts to represent null values for use in relational

databases includes descriptions of operations for these representations. There are several authors who have conducted research in this area, producing a series of articles dealing with their findings.

Codd introduces a null substitution principle and maybe operations [Cod75], in a discussion of relations. In a later paper, [Cod79], he extends the relational model with the inclusion of nulls to improve the semantics of the model. He speaks of atomic and molecular semantics, and attempts to generally improve semantic representation by defining the semantic aspects of relationships with the use of property-relations. He also models associations, generalizations, and event precedence with these definitions.

Grant's series of articles [Gra77,Gra79,Gra80] begins with a note on the inconsistency of Codd's maybe operations. He goes on to describe a method of representing partial values as ranges or sets of possible values. Descriptions of various relational algebra operations are given. He also covers the notion of functional dependencies for partial values. His operational definitions rely on the introduction of a maybe-equality.

Biskup uses the existential quantifier to represent null or incompletely known values in database relations [Bis81]. His view is that a relational tuple

denotes a statement of predicate logic. He also incorporates the universal quantifier to reduce redundancy. Using these quantified variables, Biskup extends the basic relational algebraic operations to relations with null values. The actual range of values is not used to infer information. Marked existential quantifiers may allow the inference of more information. In a later paper [Bis83], he uses classical notions of predicate logic to elaborate Codd's maybe tuples, and Reiter's closed world concept. In this paper, he introduces the "appropriate scheme assumption", which can describe an appropriate state of reality for which information is incomplete. This assumption is simply that the chosen scheme is an appropriate representation of the modeled world. This, in turn, leads to an "incomplete information assumption", which means that the modeled world is not completely specified--an open world view. This is used to develop a model of a system containing incomplete information. using sets to deal with redundancy and duplicate removal (comparison of information content). Included are algorithms for computing these extended operations.

Lipski investigates incomplete information systems by defining objects with properties which coincides with a "description language." His tools are mainly those of logic. This system is modeled in terms of equivalence classes. He introduces additive and

multiplicative normal form to be used in determining this equivalence. These forms, respectively, are conjunctive and disjunctive forms of queries and are further defined in chapter two. Incomplete information (Codd's description) is extended to the case when the subset representing this information is the whole domain [Lip79]. In this paper, a simple query language is introduced which emphasizes the distinction between the external and internal interpretation of information. Equivalent transformation of queries is discussed to provide the semantically correct evaluation of queries. Similar issues are presented in yet another paper [Lip81], with the introduction of a complete axiom system for internally equivalent transformation of terms, and a method for computing the internal interpretation of arbitrary terms and a broad class of formulas. These algorithms may be exponential in the worse case, but are of practical use in evaluating real-world queries. This later approach deals mainly with topological algebras.

Imielinski and Lipski discuss null values in the context of semantically correct processing. This approach is similar to the external interpretations and lower values defined in [Lip79]. A later paper introduces two semantically meaningful extensions for operations on tables with nulls of various kinds. One is based on Codd's usual nulls, and the second is based on

marked nulls. Marked nulls incorporate a scheme to differentiate one null from another. In this way, a null value in one relation can be associated with a null value in another relation, in the situation where they actually represent the same partially known information. A third theoretical system which can handle difference operations under a closed world assumption, is based on a device called a conditional table. It is mainly of interest in the proof of correctness of the second system mentioned. Definitions of Codd's multitables, representational equivalence, and valuation tables are developed. Representational equivalence concerns the manner in which the equivalence of nulls in different tuples can be determined. A valuation table is a relational table in which all nulls are replaced by variables, a variation on the marked null. A conditional table is an extension of a valuation table, by specifying conditions associated with each null present. In a later paper, the influence of dependencies on the processing of tables with null values is examined [Imi83].

Many sources offered suggestions for representations of nulls, or possible extensions which can be incorporated into already existing theories. Bowers [Bow84] suggests a method for storing aggregate-incomplete data, and reconciling these with the set of non-null data values. Wong [Won82] uses statistical

properties in an attempt to deduce some knowledge from null values.

Goldstein [Gol81] deals with constraints on the appearance of null values which denote missing values, and are termed disjunctive existence constraints. He attempts to show that no set of disjunctive existence constraints will properly model all constraints unless the universal instance assumption is in effect.

Lien [Lie79] considers null values in the framework of multi-valued dependencies, which affect the treatment of such values during processing. He demonstrates that multi-valued dependencies can be modified to allow inclusion of nulls in the universal relation.

Osborn [Osb81] presents algorithms which perform insertions in a multirelation database allowing no nulls, marked nulls, or unmarked nulls. The conclusion reached is in terms of the tradeoff between a universal instance with no nulls, which allows less data to be stored, and a universal instance with marked nulls, which is more costly to maintain.

Sagiv [Sag81] states that with a pure universal instance assumption, nulls must be allowed. He allows for a universal relation scheme, defining a representative instance to determine if the functional dependencies are satisfied by the database. Null values exist

only in the representative instance in this case. A modified foreign-key constraint is proposed as a possible guarantee of lossless joins. Join dependencies are considered with respect to null values.

Sciore [Sci81] deals mainly with placeholder nulls. He considers an open world assumption necessary for a missing value-type null, and a closed world assumption sufficient for a non-existant value null. In his representation, an object is a means of specifying which tuples (or portions of tuples) in a universal relation correspond to facts. The facts known about the world being modeled determine the allowable patterns of null values. The set of objects are seen as the semantic constraints on the representation. He assumes that the null set is always present as an object. His approach allows subsumption among tuples.

Vassiliou [Vas79] extends functional dependency to apply to missing value nulls. Operations on domains including these nulls exhibit different behaviors and must therefore be examined. He suggests the substitution of values from the domain to test for inconsistency or contradiction. He also describes a many-valued logic approach with examples using denotational semantics to interpret the problems stemming from the inclusion of null values [Vas80]. Treating queries as continuous functions, he presents an algorithm for

evaluation of simple queries to illustrate these acceptable semantic interpretations.

Siklossy [Sik81] describes an efficient algorithm for query evaluation which requires no Boolean normalization nor case analysis and value substitution for missing values. This algorithm, adapted from Vassiliou's, does not compute principal normal forms, and substitutes primitive terms for query simplification.

Zaniolo [Zan82] uses a "no information" null value. which leads to null tuples and extended relations in the representation of relations containing these nulls.

The Areas of Needed Research

Many of the authors discussed various topics which they feel require more extensive study. These topics deal mainly with continuations of the particular viewpoint of the author in question. All of the suggestions are related to the semantic interpretation of queries on databases with incomplete information.

Buckles speaks of the need for developing a nonprocedural query language. to allow a more effective interface between a user and a database which contains

imprecise data. Along these same lines, he suggests the personalization of query responses, dependent upon the access view of the user. A database which currently supports "views" should easily accomodate such an addition, if the interpretation of incomplete information is sound. Buckles also believes more study should be directed toward the area of fuzzy functional dependencies between domains. McDermott suggests a study on the relationship between the logic of non-monotonic reasoning and that of incomplete information.

Imielinski is concerned that functional dependencies require special consideration when null values are allowed. Further investigation involving the inclusion of nulls in specified fields (e.g., null key attribute values) is suggested. Sagiv believes that functional dependencies can by correctly dealt with by adding an extended join dependency, to correctly interpret a representative instance of a database. Osborn feels that some reasonable restrictions on functional dependencies would yield more efficient algorithms for evaluation of queries.

Several authors discussed the availability of a limited amount of knowledge which may be inferred from partial information. Wong believes that the possibility of using inconsistent information will result from finding a reliable method of representing such nulls.

Winograd sees memory representation and accessing strategies as the key in finding useful formal characterizations of non-deductive inference modes. Biskup introduces the possibility of using indexed nulls (modified from the use of Skolem constants for indexing) to allow inference to be taken from incomplete information. He notes that a difficulty which may arise from this is the introduction of functionalities which are nonexistant. Another problem is related to the operations of difference and division, where negative information is required. His assumptions of knowledge concerning this negative information are too weak to preserve the image of the relation.

Lipski mentions the need to investigate the logic involved in the internal interpretation of formulas contained in queries. He questions the decidability of the logic of molecular formulas, because he sees a need to develop a simple axiom system for such a logic. Vassiliou feels that a study of the semantics of acquisition of information by a database is necessary. He speaks of internal acquisition as the non-ambiguous substitution of null values, and external acquisition as modification operations specified by users of the database. Siklossy sites the need to investigate proof procedure computation versus case analysis computation in particular domains to insure semantically correct and efficient processing.

Proposed Area of Study

Although many algebras for dealing with relations containing incomplete information have been developed, there is still a need to examine some set of algebraic operations from the standpoint of the internal interpretation of the operations. This internal interpretation is the actual semantic meaning given to the information and determines how that information is used in the processing of a query on a given instance of a database. If the differing interpretations of these values can be brought more closely in line with each other, the difficulty of finding a correct method of processing will diminish.

After examining the existing algebras, and their semantic interpretations, which have been found in the literature, a set of sound operations is proposed. These may be applied to a relational database to allow null values representing incomplete information. These operations will allow the semantically correct evaluation of these specifically defined null values, omega-1 and omega-2. This will be of use in the development of a relational database which will accept partial information for storage, and incorporate this incomplete data in the evaluation of queries. In addition, it appears that the inclusion of partial information could be of great importance in the modeling of realistic

situations where response to query evaluation determines future action. In particular, a decision support system which relies on uncertain information to produce evaluations would benefit from the ability to represent and process uncertain information in a well-defined manner.

Chapter Two : Dealing with Null Values

The fact that so many types of nulls have been distinguished, and the amount of existing research in the area of representing nulls, gives the impression that little is left to be done. Various problems still exist. Actually, although much has been written about the problem, no concrete solutions have been developed for the many problems of representing nulls for use in the retrieval of data from databases. This stems from the different approaches which can be seen in the literature. Much research is shown to be so computationally complex that it is only of theoretical value. Similarly, the simpler representations exhibit complexity and ambiguity regarding interpretation.

One reason for the distinct differences in approaching the problem of nulls is the view of the modeled world that the database designer choses. The open world view demands the allowance of null values. If this view is adopted, the known states are finite, yet may be incomplete, thus allowing inapplicable or unknown values to be represented. The closed world view is more restricted in that its knowledge is complete, and therefore its states are infinite. This view lends itself to a more restricted modeling of the

world, with results which can be anticipated and strictly proven. These differing views have lead to the development of various suggested representations and algebras to deal with null values. The more prominent ones are discussed in the following sections.

Existing Algebras

The studies which have been conducted previously may be approximately ordered in terms of existence and influence on subsequent research. Codd's introduction of three-valued logic to deal with null values in relational tables is apparently the first published effort in this area. However, Jaegermann was working with incomplete information systems in approximately the same time frame. Subsequent research from these introductory studies can be split into two distinct approaches.

Biskup and Grant have built their work upon Codd's foundation. Lipski, initially alone, and a bit later, with Imielinski, adopted Jaegermann's study as the supporting background for their work. Vassiliou follows this second approach, but leans more toward defining the underlying semantics, rather than attempting to define an algebraic approach to relational operations. His reasoning for this approach

is that the semantic interpretation of the incomplete information should determine the way in which the information is processed.

The two approaches can be compared in many respects, yet each retains a distinct methodology which demands investigation as to compatibility with the other. A brief synopsis of the works of Codd, Biskup, Grant, Imielinski, Lipski, and Vassiliou is presented in the following sections, to provide insight as a base from which a further study may be conducted.

Codd's Original Work

Codd uses the null value exclusively as a place-holder null, i.e., value at present unknown. He defines a null substitution principle which simply states that the null may take on a value in a finitely restricted attribute domain. This principle is used in determining results of comparisons between null and non-null values in the database. Codd's truth tables are shown in Figure 2.1. [Cod75] An example of the join operation is shown in Figure 2.2. The maybe join is symbolized by "=@".

AND	T	F	@
T	T	F	@
F	F	F	F
@	@	F	@

OR	T	F	@
T	T	T	T
F	T	F	@
@	T	@	@

@ = unknown

NOT @ = @

Figure 2.1. Codd's truth tables
for the omega null.

R

A	B
u	@
@	2
w	1

S

C
@
2

R[B=C]S

A	B	C
@	2	2

R[B=@C]S

A	B	C
u	@	@
u	@	2
@	2	@
w	1	@

Figure 2.2. True and Maybe theta join.

His approach has been highly criticized because the three-valued logic presented has been found to be non-truth functional. (Refer to the next section.)

In a later paper, he has introduced special objects, as sets of n-ary relations, and the algebraic

operations on these sets. For example, an entity relation would list all the entities of one type that currently are contained in the database; a property graph relation contains property types assocoiated with entity relations. He forms a graphic depiction of relations of entities, properties of entities, and associations of entities, which creates a domain object upon which to operate. These objects make up the model of the database.

Codd lists four personalities which a data model should have if it is to capture more semantic meaning : tabular, set-theoretic, inferential string-formula, and graph-theoretic. The tabular form is for display purposes. The set-theoretic view allows search without navigation. Inferential techniques may be applied using the third personality, which is modern predicate logic. The graphic view is useful for development and maintenance.

A Different View of Codd's Work by Grant

Grant is one of the first to point out the previous problem with Codd's methodology, and seeks a solution in a slightly different representation. Codd's truth-value evaluation fails to retrieve a tuple containing a null value when a query contains mutually

exclusive terms dealing with that attribute. For example, if a tuple contains a null value for the attribute Status, and a retrieval is attempted with a disjunctive expression containing the condition (Status = 2) and another with NOT(Status = 2), a tuple that meets all other criteria for retrieval will not be found in the result returned.

In a series of papers concerning incomplete information in databases, he introduces concepts which he feels deal more correctly with the evaluation of this information. Grant's model consists of nonatomic data values, restricted to finite ranges of domain values. In his approach, null values may be properly replaced with an actual range which is defined for the given domain by integrity constraints. His evaluation of the null value in processing exhibits the following behavior: a true result is returned if the predicate is true for all proper substitutions, and a maybe result is returned if the predicate is true for at least one proper substitution. This leads to a definition of true equality and maybe equality. He introduces three notions of operations which might be applied to his representation-- set theoretic, true, and maybe versions. His true intersection operation, for example, omits all finite range entries, while the maybe intersection deals with these ranges, selecting entries for

which the ranges overlap even slightly. Figure 2.3b shows an example of a true selection compared with a maybe selection, and the way in which the finite ranges are interpreted for this operation. [Gra80]

EMPLOYEE

E#	EAGE	ESALARY
109	35	(15000,18000)
123	(30,40)	null
250	null	20000
300	49	25000

Figure 2.3a. Grant's nonatomic data items.

Query: Select Employee where
Eage > 33 and Esalary > 19000

True Selection

300	49	25000

Maybe Selection

123	(30,40)	null
250	null	20000
300	49	25000

Figure 2.3b. Grant's True and Maybe Selection

Grant also takes into account the generalization of functional dependencies which is necessary for the partial information model with which he is working. He shows that decomposition can be done for tables with these partial values. The following definitions illustrate the interaction of Grant's range values with

functional dependencies. [Gra79]

> V is a table with columns A and B, r1 and r2 are arbitrary rows of V with values a1, a2 for A and b1, b2 for B respectively. For any two rows of V either a1 = a2 or not(a1 =m a2).
>
> (i) A --\>1 B if : A and B have single entries only and if a1=a2 then b1=b2.
> (ii) A --\>2 B if : A has single entries only and if a1=a2, then b1=b2.
> (iii) A --\>3 B if : B has single entries only and if a1=a2, then b1=b2.
> (iv) A --\>4 B if : if a1=a2, then b1=b2.

Grant also shows, in a third paper, a method of allowing substring predicates to define partial character strings. His representation no longer views elements as non-decomposable.

Biskup's Algebra

Biskup relies upon logical quantifiers to give meaning to the two types of nulls under study. The existential quantifier "∃" is used to denote missing information; a value exists, but is unknown. The universal quantifier "∀" is used to represent the fact that for all values in the domain, the information in question is valid. This is of use in removing redundancy from a relation. The quantifiers are added to

the set of regular values allowed for attributes. A value-relation is necessary for this representation (see Figure 2.4), adding a range declaration for every relation.

A	B	C
∃	a2	a3
a1	∃	∀
a2	a1	a2

with {a1,a2,a3,a4} as
the value-relation

Figure 2.4. Biskup's logical quantifiers.

He defines a partial ordering relation as an equivalence relation which is developed for computational reasons. Redundancy is eliminated by using this equivalence relation which is induced by the partial ordering. Information viewed in this manner forms a distributive lattice.

In a subsequent paper, Biskup introduces different rules for data extraction, by defining a three-valued logic, and for duplicate removal, considering one null the same as any other. Biskup notes that when computing a representative relation (removal of redundant tuples), it is possible to lose

information concerning ranges. His method for dealing with comparisons of tuples is shown in Figure 2.5.

v(C) \ v(D)	∃	known value	∀
∃	-	-	+v(D) <-- ∃
known value	-	if v(C)= v(D):- if v(C)= v(D):+	+v(D) <--v(C)
∀	+ v(C) <-- ∃	+ v(C) <-- v(D)	unknown range: + v(C) <-- v(D) <-- ∃ range defined: + for all x in range, v(C) <--v(D)<-- x

no contribution is indicated by "-"
modification is indicated by "+"

Figure 2.5. Biskup's table for tuple comparison.

He extends the relational table to include a two-valued tag field, which he calls "Status", to be stored with every internal tuple. Status may be either definite or maybe. and is used to determine the way in which a tuple is processed in any of the basic operations. A tuple with a definite status expresses a statement which is true of the model. Further, any tuple of the model is obtainable from a stored definite tuple via substitution. Biskup requires

that a weak minimality must be somehow related to the internal representation of a relation, thus preventing totally unrelated tuples to be stored. An example of his tables and his extended natural join is shown in Figure 2.6. [Bis83] A Status of "D" means that the information contained in the tuple is completely known; a Status of "M" signifies incomplete information.

R

A	B	C	Status
a	b	c	D
@	@	c	D

S

B	C	D	Status
b	c	d	D
e	@	d	D

R|X|S

A	B	C	D	Status
a	b	o	d	D
@	b	o	d	M
@	e	c	d	M

Figure 2.6. Biskup's relations with Status attribute.

Concerning information content, Biskup defines equivalence with the notion of weaker and strictly weaker orderings. Redundancy may be m-redundant, md-redundant, or strongly m-redundant. A formal definition of these term follows:

For a given tuple r, and element of an internal relation R:

r is m-redundant in R, iff r is an element of the set of maybe tuples of R, and there exists a tuple s, not equal to r and weaker than r.

r is md-redundant in R, iff r is an element of the set of definite tuples of R, and there exists tuple s in R, and a definite tuple t in R, such that r is not equal to s or t, and s is weaker than r, and r is weaker than t.

r is strongly md-redundant in R, iff r is md-redundant in R, and there exists an attribute A in R such that s(A) is not equal to r(A).

His relational algebra operations are proven to be restricted and adequate. [Bis83]

The Second Approach to Representing Nulls

Lipski is responsible for the underlying study which has influenced both Imielinski and Vassiliou. His mathematical theory is intended as a logical background for studying the problems connected with incomplete information. He views an information system much as Jaegermann, in that the system stores information (which may be incomplete) concerning properties of some objects. These objects are mapped, in an incomplete way, into the system. He states that the representation of partial information

is not expressible in an approach based only on null values. Two extremes of representation are available with this self-limiting approach-- either everything is known or nothing is known.

To allow incomplete knowledge to be represented, the value of an attribute for any particular object must be represented as a function. For a distinct attribute value to be known, a function must be defined from an object (defined in terms of attribute domains) to the specific attribute. A partial ordering of values may result in a complete extension, termed a completion.

One of Lipski's main concerns is the difference between the external and internal interpretation of the system's information about the world. A database itself has two conceptual interpretations. All of the information which is actually contained in the database, and its meaning concerning the world the database models, comprises the internal interpretation of the database. The external interpretation is the real world modeled by the system in an incomplete way. It is obvious that these two interpretations could differ widely. For example, a system contains information concerning objects a, b, c, and d, and the set of objects known to be red consists of object a, while the set of objects known not to be red is only d. It is not known whether

or not b and c are red. The external interpretation of a query concerning red objects must deal with the set of all objects which are in reality red--{a}, {a, b}, {a, c}, or {a, b, c}. But the system does not contain sufficient information to exactly determine this set. The lower bound on a query "red" is {a} which must be contained in the external interpretation. The upper bound is {a, b, c}, and this set can not be ruled out as possibly belonging to the external interpretation of the query. [Lip79]

Lipski introduces a method for establishing a query language which takes into account the internal interpretation of the information available. This two level language is made up of terms, which are subsets of the set of objects, and formulas, which are built from terms and express some fact concerning the system. The value of a query in a complete system may differ from the value when the internal interpretation is taken into account. The internal interpretation of a query is equal to the external only when the bounds of the complete system are explicitly known. The following example illustrates the difference between the two interpretations. [Lip79]

Assume the information:

possibly white objects	a, b, c, d
possibly black objects	c, d, e, f
possibly red objects	b, c, f, g

The interpretation of "White OR Black" is:

{a, d, e}	external GLB
{a, b, c, d, e, f}	external LUB
{a, e}	internal

where GLB = greatest lower bound
LUB = least upper bound

Primitive terms are defined as products of values from a given attribute domain. Coprimitive terms are summations of values. Primitive and coprimitive terms are used to arrive at additive (summations of primitive terms) and multiplicative (products of coprimitive terms) normal form. He presents a method for constructing these forms from queries. [Lip79] Below, an example of the forms is shown.

For the following query:

<Dept# in (2,3)> *
-<Sal < 10000> *
<Hireyear > 72> +
<Age > 50> * <Sal < 15000>

Figure 2.7a. Example query.

Multiplicative Normal Form

(<Dept# in (2,3)>) *
(<Hireyear in (70,71,72)>
+ <Sal >= 10000>) *
(<Age <= 50> + <Sal >= 15000>)

Figure 2.7b. Lipski's MNF.

Additive Normal Form

<Age <= 50> * <Dept# in (2,3)>
* <Sal >= 10000> + <Dept# in
(2,3)> * <Sal >= 15000> +
<Age <= 50> * <Dept# in (2,3)>
* <Hireyear in (70,71,72)> +
<Dept# in (2,3)> * <Hireyear in
(70,71,72)> * <Sal >= 15000>

Figure 2.7c. Lipski's ANF.

Transformation of queries into either of these normal forms is complicated by the classical problem of the minimization of boolean functions. Lipski adds two operators to allow for the possibility of incomplete information. Surely and possibly operators, respectively, represent the least upper bound and the greatest lower bound of the available information. Lipski also notes that if intervals are allowed, some classes of information are not representable.

Lipski continues with his investigation to find that internal equivalence is decidably stronger than external equivalence. His theory is further refined to

show that a topological Boolean algebra has an analagous role with respect to the internal interpretation as that of a Boolean algebra with respect to an external interpretation. By indirectly defining internal properties of objects, possible processes of increasing knowledge may be specified. A system of distinct representatives may be formed, which provides a possible completion of the information concerning objects which result in a true value for a given formula.

Special terms are defined as the internal interpretation of terms including a broader class of terms containing a new unary operator, "surely". These special terms allow weak additive and weak multiplicative normal forms. Special formulas are finite disjunctions of elementary formulas and are similarly defined by Special Disjunctive Normal Form. SDNF is achieved by transforming a formula into its externally equivalent form from either the ANF or MNF of the query.

In Lipski's most recent effort, he studies the interaction of dependencies with null values. He uses the well known chase procedure to transform dependency information for a table into an equivalent table with null values. With arbitrary implicational dependencies, the transformation is not quite equivalent, but

using only the operations of projection, positive selection, union, natural join, and renaming of attributes, any corruption introduced in not detected in the result of the query.

Extensions of Lipski's Theory

Imielinski, with Lipski, attempts to define precise conditions for meaningful extensions to operations for tables containing null values. He shows the result of Codd's null in projection and selection, and a marked null used in projection, positive selection, union, and renaming of attributes. No form of the universal relation assumption is required for these operations.

Imielinski formulates conditions to be embodied into the definition of a representation system. He introduces Codd tables, Valuation tables, and finally, Conditional tables. Codd tables are the usual relational tableau which contain omega to represent a null value. The Valuation table is simply a Codd table in which any occurrence of a null may be replaced with a variable, the same variable being used for a specific attribute. Conditional tables are Valuation tables with an added attribute, a condition (reminiscent of Biskup's status attribute) which may be used to con-

strain the values in the tuple. An example of a conditional table is given in Figure 2.8.

A	B	C	cond
a	b	z	(z≠c)
a	y	c	(y≠b)
x	b	c	(x≠a)

Figure 2.8. Imielinski's Conditional table.

These representations are intended to enable correct evaluation of relational expressions, instead of single operations.

A Denotational Semantics Approach

Vassiliou defines two specific nulls, omega, inapplicable, and theta, missing, in his approach. He views the information in a database as a finite approximation of the real world, which requires infinite information for complete modeling. The only way to model this infinite information is to allow an inconsistent object (one which cannot be represented in a finite way). Functional extensions between domains are defined such that they are continuous. It is also noted that the evaluation is very inefficient. Query

evaluation is defined in terms of these continuous functions between data types.

The system is viewed as a lattice with top, representing the placeholder null, nothing is known, and bottom, representing the inapplicable null, value missing. In all domains, all other elements are equally accurate, producing a flat lattice. Query transformation is simply symbolic manipulation into suitable forms for evaluation. The disadvantage in this representation is that any algorithm must deal with normal forms of a length that is exponential in comparison to the number of primitive terms.

A later study by Vassiliou extends functional dependency theory to handle nulls. Data dependencies are purely syntactic notions introduced to capture semantic information in a relational database, predicates on instances of the set of relations. A null equality constraint is introduced which determines equivalence classes for null values. This constraint states simply that two null values must take on the same value in any substitution. This leads to the fact that if the evaluation of a function (operation on a domain) returns the same results with different substitutions, the incomplete information is not essential for the evaluation. He gives an example in which the functional dependencies are not satisfied because of a

null value. The following relation with nulls illustrates how a tuple might violate a functional dependency. The FD for the relation R(ABC) is AB-->C.

R

A	B	C
a1	b1	-
a2	b2	c2
a3	-	c3
a1	b1	c2

Figure 2.9. Vassiliou's placeholder null.

It is apparent that the first tuple in R is in violation of the functional dependency AB-->C because the fourth tuple contains no nulls and is the same for attributes A and B as the first tuple. Therefore, the first tuple must not be contained in R.

Vassiliou takes the view that databases are often overconstrained because incomplete information is not allowed to exist in the system. He feels that the notion of weak satisfiability and allowing nulls will enable a weaker set of constraints to be valid in more instances, while still retaining semantic integrity.

Areas of Further Research

Biskup lists the following problems which he feels are still open and in need of further research [Bis83]:

(1) A clear formal semantic of the null value is missing.

(2) In the absence of a formal semantic, the meaning of updating operations involving null values is not investigated.

(3) There is no common basis for the two different rules for processing nulls.

(4) There are some arbitrary choices in using one of the rules for a specific task.

(5) Justification of these rules on the basis of intuitive appeal and in the context of application-verifiable assumptions is missing.

(6) There is no justification, except feasibility, for considering nulls only locally, on the level of tuples.

(7) There is no analysis of whether the proposals are "best" in some sense.

(8) Although difference and division are treated, there is no discussion of the underlying view of negative information.

(9) There are no proposals of how to subsequently treat tuples of a "maybe" result.

He addresses each of these issues in his later paper. However, it is apparent that there are many areas which are inadequately investigated. Many of the problems which have developed from the prior research will be discussed in the following chapter.

Chapter Three : Algebraic Solutions

In an effort to clarify the existing problems, this chapter deals with the existing algebraic methodologies proposed in terms of their interpretation of the world being modeled, the operators supported in the algebra, the efficiency and usefulness of the algebra, and the possibilities for further investigation pointed out by the researchers themselves. We view each of the approaches in the framework of the relational database model, moving from the basic representation of null values to their effect upon functional dependency theory, following the lead of the more prominent researchers on this subject. From Codd's original introductory discussion of the inclusion of nulls, we can trace the evolution of research and the interaction which has occurred among those searching for a solution to the problem of nulls.

The Meaning of Null Values

Codd bases his relational model with nulls on the open world assumption. To support this view, range declarations are necessary for each attribute domain in the model. His null substitution principle is useful

only with known, finite domains. His model rests within the relational database model in that he restricts all entries to atomic values.

Grant was the first to point out the non-truth functionality in Codd's representation. His representation allows partial values as elements which are restricted as to information type. These partial values are introduced as ranges within the domain. This, unfortunately, requires that the relational model be modified to deal with non-atomic data elements-- intervals, in this case. He has used numeric intervals almost exclusively, the exception being an example of substring capability in his most recent article. He also allows the all purpose placeholder null to be an element in his tables.

Biskup assumes, in general, an open world assumption by requiring range declarations for attributes in relations. He notes that a closed world interpretation is possible for negative information by appropriate substitution of nulls, but that if an all-null tuple is stored for any relation, the interpretation is open even for negative information. His introduction of a status attribute for each tuple is interpreted as closed world for every definite tuple in the system. He points out that his system deals correctly with

negative information, a failing in other systems. His appropriate scheme assumption and incomplete information assumption are required, however. His method of representing nulls is based upon the storage of statements concerning the modeled world, not the relations.

Lipski begins by defining a minimal system, one which contains no information at all. Using the intuitive definition of extension, he discusses methods of increasing the information about an object. These methods concern determining the upper and lower value of a term describing an object. He states that there is no inductive method for this, but that an equivalent term in additive normal form can be used to determine the upper bound, and multiplicative normal form to determine the lower bound.

He differentiates between a complete system, one which has a unique value for every attribute of every object, and a complete extension, termed a completion, which can be derived from an extension containing incomplete information. Descriptors are treated as nondecomposable elements without any internal structure. Lipski's model, if its information is complete, directly corresponds to the relational model.

No form of the universal relation assumption is required for Imielinski's representation of null values. An open world interpretation is used for these

tables, which means they are not able to represent negative information.

Vassiliou's main contribution is the formalization of null values in the framework of denotational semantics. He believes that there are alternatives which exist in the semantic interpretation of nulls. The universal relation assumption is used, as it must be in any model which relies upon the dependency theory of relational systems.

Representing Null Values

Codd's intent is to incorporate semantic data modeling to support and define null values. He states that his discussion should be regarded as preliminary and in need of further research. One of the concerns about the method of representation is that two different rules are used for processing null values. For duplicate removal, all null values are not recognized as equal unknown values, yet to prevent non-duplication of tuples in processing, nulls are treated as the same value.

Grant allows duplicate rows (entries) to be present, for although an interval in one row may be the same as an interval in another row, the actual values they represent may be different values. His interpre-

tations of subsets and equality is unlike Codd's. With two columns such as R = [null;1] and S = [null;1;2], Grant allows that S is a "maybe subset" of R because of the possibility that f(2) could be null. This leads to a definition of true intersection in which all non-single entries are omitted. This constraint is the basis for the difference in each of his true versus maybe operations. Some solutions that Grant proposes, but does not elaborate upon, include some type of coding for repeating intervals, and the restriction of these partial values to certain columns.

Some of the problems which Biskup admits with his representation are the comparison of the universal quantifier null (Ψ), the evaluation of negative information in difference and division operations, and the additional complexity involved in using range declarations when trying to extract information. Additional cost is generated for elimination of duplicate and redundant tuple and also for the storage of maybe tuples. His extended operations may themselves produce duplication. He considers null values in tuples only locally.

Some of the problems which Lipski has suggested concern the representation of the incomplete information. Admissible subsets of domains determine whether or not the information can be modeled with an incom-

plete relational database. He notes that if ranges are used, some information can not be represented. For example, how can the information that some measurement is not in the interval (50,100) be stored? Another problem stems from the fact that a completion may have nothing to do with the real process of increasing a system's information. Many of the algorithms used are of considerable computational complexity.

All of these problems seem to suggest that the devices suitable for representing incomplete information depend upon what processing needs to be correctly performed. This realization led Imielinski and Lipski to introduce the conditional table, which restricts possible values of a null by explicit conditions stored in the system. It was developed as a theoretical system, mainly to support the V-table.

Vassiliou believes that any occurrence of the inconsistent null should cause rejection of the containing tuple from either the true or maybe result of a query. He disallows queries involving the unknown null, suggesting that "nothing" does not exist as a value. He shows examples in which it is noted that Codd's rules are sound but not complete.

Operations on Tuples Containing Nulls

The operators which Codd defines for his model include union, intersection, difference, Cartesian product, and projection (using the non-duplication rule). Further investigation has lead him to define true and maybe theta joins, true and maybe equi-joins, and theta selection. In addition, he introduces operations for non-union compatible tables such as outer union, outer intersection, outer difference, outer theta join (which can generate nulls), and outer natural join. He notes that natural and equi-joins will lose information when there is a nonequal projection on the join attribute. Generated null values are interpreted as unknown values if an open world interpretation is in effect, but as an inapplicable property with a closed world assumption.

Another problem, which has been pointed out by many others, is that a query such as EMPLOYEE[Age<=50 U Age>50] does not yield every employee as should be expected. This is a result of the manner in which Codd's truth tables function. He attempts to rectify the problem with the introduction of the maybe operations. This in itself places an added burden of knowledge upon a user of this type of system.

Grant notes that it is not, in general, possible to obtain the resultant table by expanding a table to all possible tables (using a null substitution

principle for the interval), and then applying whatever corresponding operation is requested. This is a problem because substituting for nulls may violate an integrity constraint. For example, a maybe join may prove lossy. Again, this requires a user to be more knowledgeable about the operations and the underlying semantics of the system.

The operations which Biskup defines for his model include an extended natural join, extended projection, extended selection, extended comparison, extended union, extended difference, extended division, and extended update. The "extended" concept refers to the additional status column, and the manner in which it is used in the processing.

In an attempt to minimize the expense of formulating a completion during processing, Lipski uses query transformation to arrive at an externally equivalent form for the query. He believes that this external interpretation is sufficiently appropriate for a naive user who is unaware that the system may contain incomplete information. In other words, to use the internal interpretation of the incomplete information, the user must understand that it may in fact exist in the system, and that it may affect the results of queries upon the system. He also suggests that binary descriptors be used to define a numeric property which he calls

"known", to aid in the semantically correct interpretation of possibly incomplete information.

Lipski shows that while two queries can be shown to be externally equivalent for a complete system, there is no known method for computing the lower bound of a formula for an incomplete system. He proposes that a subclass of queries be developed, stating that most query languages are more expressive than necessary. As an example of the complexity of these algorithms, one of the examples given in his discussion does not follow the transformation rules in an attempt to be more efficient [Lip79].

Special disjunctive normal form is defined as a method for transformation of special formulas. Lipski briefly discusses the fact that the length of the transformed query in SDNF grows exponentially with the length of the original query. He does temper this information with the assurance that a normal query will very seldom reach such a length as to make the transformation unfeasable. The introduction of the unary operator "possibly", which may be used in these special formula. is the culprit here, as the method of evaluation must enumerate all completions of the object in question.

Imielinski (with Lipski) has extended Lipski's work to define the operations of projection, selection,

union, and join on union compatible relational tables. He states that a relational algebra can be embedded into a cylindric set algebra to deal with incomplete information. Unfortunately, this leads to infinite relations, and creates problems concerning finite representability. His main work is concerned with extending the relational algebra for the systems he discusses, which include Codd tables, variable tables, and conditional tables. He points out some of the problems with these systems as he progressively seeks to alleviate them with more correct representations.

Codd tables correctly support projection, selection and union on independent attributes, but cannot handle join operations. Biskup's join is correct only if the two relations are independent, with correct results for other simple relational operators. Imielinski and Lipski use directed graphs to illustrate the workings of their operators. The attribute domains are infinite, and no disjunction may contain both an equality and a negation of the same equality. This seems to be the same qualification for the correct evaluation of Biskup's join. A closed world interpretation of these tables suggests that negative information can be represented. This is determined by the use of the null substitution principle developed by Codd. V-tables are not a representation system, and therefore cannot support projection-selection expressions. The infinite

attribute domains prevent reasonable use of the null substitution principle. V-tables can handle arbitrary conjunctions of atomic relational formula containing constants or variables.

Vassiliou's theorems are proven with simple expressions involving only equalities. The restriction seen before (not allowing p and not p in a series of terms) is present in Vassiliou's work, also. Because of the denotational semantic approach, there is the requirement of complete lattices and continuous functions. The algorithms are transformed into a propositional calculus form, either principle disjunctive or principle conjunctive normal form, which may not always be applicable.

Domains are finite and known to the system. Only two attribute relations are considered. One concern in this manner of evaluation is in keeping a dependency true while substituting values of the domain for a null value. This is the only method to determine a "false" tuple. This is, however, a very difficult procedure. Vassiliou states that it is natural to weaken expectations and allow a margin of uncertainty when nulls are included. It is possible that it is better to leave the database model incomplete and not allow for substitution of null values.

Functional Dependency Theory and Null Values

The modification to the relational model forces Grant to deal with dependency theory, which is done by interpreting only the simplest form of dependency statements required for the model. Because non-atomic elements are present, first normal form is not attainable. Grant redefines functional dependency for these tables with partial values. The problem of negative information determination is side-stepped by building negation into the primitive operators that are defined to work on these tables. He requires union compatibility for operations involving two or more tables, and states that the obvious definitions for these operations are counter- intuitive.

Grant's definition of true and maybe functional dependencies is used to define decomposition for relations with intervals. He suggests that some warning mechanism be incorporated into the processing when attempting to satisfy integrity constraints. He notes that an interval may have to be changed during processing for the same satisfaction.

Imielinski shows that V-tables are capable of representing functional dependencies and join dependencies. They correctly support projection, positive selection, union, and join.

In this most recent discussion concerning view dependencies, the universal instance assumption is seen as a sequence of projections. They examine the global problem of incorporating both implicational and inclusion dependencies in the table content. They note the the problem could be formulated more generally as the problem of proportion between so-called extension and intension of a database. The extension is the current state of information contained in the database. The intension represents time independent properties of the information which may be contained at any time in the database.

Unlike Codd and Lipski, who consider only the retrieval aspects, Vassiliou incorporates dependency theory in his investigation. Vassiliou's interpretation of functional dependencies is as predicates on instances of a relation. He notes that the inconsistent null cannot be present in a database where certain semantic rules are required to be valid. The evaluation of a tuple containing a null value by using substitution of each domain value is unacceptably complex. The algorithms suggested for transformation of queries make this substitution unnecessary.

The Development Of The New Approach

The non-truth functionality of Codd's original approach has been used as a starting point for much of the ongoing research in this area. The complexity of the developed methods for dealing with nulls also originates from this approach. Developing a truth functional representation of null values and processing techniques for tables containing nulls is the goal of the next chapter.

CHAPTER FOUR : Interpretation of Omega-1 and Omega-2

The omega null, which now shall be called omega-1, will continue to represent unknown or partially known information content. In this respect, it can be thought of as Biskup defines this null-- there exists some value in the attribute domain, but it is not known at this time which specific value it represents. Because of this interpretation, and the previously existing operations defined for omega-1, the operations defined for omega-1 in the following section will be based upon Codd's work.

The theta null, which now shall be called omega-2, is discussed most thoroughly in Vassiliou's work on incomplete information. The interpretation of omega-2 will differ somewhat from the original meaning given the theta null. Omega-2 will represent an inapplicable value. It is noted that at the same time omega-2 nulls are allowed as values of an attribute domain, integrity constraints may be somewhat relaxed and a reduction of redundant information will occur.

A more detailed explanation, along with rules, examples, and tables, constitutes the main part of this chapter. The relational algebra operations can not simply be replaced by extensions which deal with nulls.

New operations must be defined which will correctly return the expected results. Operations for omega-1 and omega-2 nulls are introduced in this chapter. Relations allowing both nulls are considered, along with a discussion of the interaction between the two in various operations.

Revision of Codd's Operations

As stated previously, the omega-1 null will be interpreted semantically as incomplete information, the existential quantifier--"there exists". One of the goals of this representation is to allow all possibly correct information to be returned using some minimal processing technique. Codd's operations will be the basis of these extended operations because of the simplistic approach (in terms of processing), and the ability to derive the desired results (in terms of possibly correct responses). The truth tables for the three-valued logic containing omega-1 are shown in Figures 4.1a and 4.1b.

AND	T	F	w1
T	T	F	T
F	F	F	F
w1	T	F	T

Figure 4.1a. Omega-1 AND truth table.

OR	T	F	w1
T	T	T	T
F	T	F	T
w1	T	T	T

NOT(w1) = T

Figure 4.1b. Omega-1 OR truth table.

A modification of Codd's null substitution principle is used to derive the values in the above truth tables. This modification relies on part one of this principle.

> Principle 1: Any expression containing an omega-1 may be replaced with an expression which contains a value for that omega-1 so as to yield a true result for the expression.

This does not negate the fact that an occurrence of omega-1 can be replaced by a value which yields a false result for the expression. By defining omega-1 in this

way, it can be seen that the truth tables which direct processing will retain truth functionality.

Codd's True operations [Cod79] work on relational tables which contain nulls in such a way that tuples containing nulls are excluded from the result. The True operations return only those tuples for which the expression evaluates to true. The Maybe operations are defined in such a way that only those tuples for which the expression evaluates to null are returned.

For the purpose of this study, and using the modified semantic interpretation of omega-1, it is desirable for the extended operations to return all possibly correct tuples. Therefore, these operations will be defined such that the results will contain the tuples for which the expression evaluates to true. All tuples containing omega-1 as the value of an attribute will now be returned as the result of an evaluation, if that attribute is a criteria of the selection involved. Presumably, users of this database will be aware of the types of evaluations they are requesting.

Operations Involving Omega-1

The following relational tables are taken directly from Codd's work on null values [Cod79]. The results of a union and difference, using relations R and S are

shown in Figure 4.2b. The same results are derived using the definition stated for omega-1 (w1).

R

w1	w1
u	w1
u	1
w1	1

S

w1	w1
u	2
u	1

Figure 4.2a. Relations containing omega-1.

R U S

w1	w1
u	w1
u	1
w1	1
u	2

R - S

w1	1
u	w1

Figure 4.2b. Union and Difference.

The following example of selections are given to show the point at which the processing of omega-1 differs from Codd's methods for the original omega null. Notice that in the selections, every possibly

correct result is returned. Codd defines Maybe Selection in which only those tuples for which the selection criterion evaluates to omega are returned. In the first selection example, both "v" and omega-1 can represent "v", thus producing the given results. Because of the interpretation that every value in the original relation for the attribute B could possibly represent the value "1", in the second example of selection, all tuples are returned.

R

A	B	C
u	w1	w1
v	1	w1
w1	w1	1
x	1	w1
y	w1	1

Figure 4.3a. Relation R.

Select A = v [R]

v	1	w1
w1	w1	1

Select B = 1 [R]

u	w1	w1
v	1	w1
w1	w1	1
x	1	w1
y	w1	1

Figure 4.3b. Selection operation.

Figure 4.3c shows examples of projections using R from Figure 4.3a. These are given to clarify the duplicate removal which may take place in relations containing omega-1.

R[B,C]

w1	w1
1	w1
w1	1

R[A,C]

u	w1
v	w1
x	w1
y	1
w1	1

Figure 4.3c. Projection operation.

Notice that only those tuple that exactly match for all attributes may be removed, i.e., [w1,1] is only present once in the resulting table, although it occurs twice in the original relation R.

In the following examples of the natural join, Figure 4.4a and 4.4b, it can be seen that this new interpretation of omega-1 and its subsequent processing differs from Codd's. Note that the second, fourth, and fifth tuple would not be included in Codd's definition of join because the value returned is not omega.

R

A	B
u	w1
w1	2
v	1

S

B	C
w1	1
2	2

Figure 4.4a. Relations R and S.

A	B	C
u	w1	1
u	w1	2
w1	w1	1
w1	2	2
v	w1	1

Figure 4.4b. Natural join of R and S.

It seems reasonable to assume that tuples two and four of the join actually represent the same information. The functional dependencies applicable to this relation are the only knowledge which would enable the determination of the equivalence of these tuples. Tuples one and three might also represent the same information. Again, functional dependencies need to be taken into account before a determination as to equivalence can be

made. As one goal of this representation is to return all possibly correct information to the user, duplicate removal must occur only for exactly matching tuples. The fact that unknown information is present must not be hidden by the method of processing. For this reason, any time a tuple is returned because of a match between a specific domain value and omega-1, the tuple will show its "speculative" property by the presence of omega-1 as the value of the attribute in question.

For a more concrete example, consider a relation in which items, manufacturer, and color are the attributes. The relation contains the following tuples: p1 = <1,A,red>, p2 = <1,A,white>, p3 = <1,A,w1>, p4 = <2,w1,red>, p5 = <2,w1,blue>. Does this mean that the item in p1 is the same item as that represented by p4, even though the item numbers are not equal? If these items in reality represent the same item, it is because the error was created during an update operation. The fact that the item number is most likely a key of the relation, can be used to determine that p1 and p4 do not represent different occurrences of the same information.

Although there are omega-1's present in each attribute column of R (Figure 4.4a), it is anticipated that normally at least one column (the prime attribute) would not contain any type of null value. However, a

projection of any arbitrary relation could always return such a table. The problem of dealing with tables containing nulls resulting from prior operations must also be addressed.

Explanation of the Theta Null

The constraints which are placed on a relational schema to avoid anomalies have been carefully studied by many researchers. It is apparent that many decomposition schemes introduce redundancy of information which must be stored to maintain the integrity of the constraints and the information. Studies have been done in an effort to reduce this aspect of duplication, specifically those conducted on non-first normal forms [Jae82]. By allowing the presence of omega-2's, information which would normally have to be represented elsewhere in an arbitrary schema, may instead be represented with less duplication.

It is explained using lattice theory [Vas79] that the theta null represents the top element, i.e., it is the inconsistent null, containing more information than can be represented. It seems more reasonable to allow a null value which can represent the fact that no value exists for a certain attribute. In this approach, omega-2 will represent the fact that there is no value

in the attribute domain which is correct for this tuple.

> Definition of omega-2: If Z is a column of R(X,Y,Z), and there is a tuple t in R such that t[Z] = w2, ∀z in R(Z), ∄t[Z] = z.

Although a tuple with an omega-2 attribute value might appear to contain the same value as another tuple containing an omega-2, two instances of omega-2 in disjoint tuples can never be considered to represent the same value. This is basically the same problem of determining equivalence which is present in the processing of omega-1. The occurrence of omega-2's in disjoint tuples are never considered equal, but are considered equal only to omega-2.

Because of this interpretation, duplicate removal for a tuple containing omega-2 is treated as normal duplicate removal. Omega-2 is a distinct value which is the same distinct value in a tuple where all other attribute values are equal.

Vassiliou states that a query involving theta (e.g., selection on the condition that something equals theta) is not allowed. As omega-2 is considered a distinct value, it seems appropriate that a query could be allowed on that value. For example, it could be beneficial to allow a query to determine which employees do

not have a phone, which can be determined by a selection of employees that have omega-2 as the value for their phone number.

It seems appropriate, then, that omega-2 must be included as a value in all attribute domains where it will be allowed to occur. It is not necessary or even desirable to include omega-2 in every attribute domain. Key attributes, specifically, should not be allowed to have omega-2 as a possible domain value.

Figures 4.5a and 4.5b illustrate the truth tables developed in this thesis for operations involving attributes in which omega-2 is an acceptable value.

AND	T	F	w2
T	T	F	F
F	F	F	F
w2	F	F	F

Figure 4.5a. Omega-2 AND truth table.

OR	T	F	w2
T	T	T	T
F	T	F	F
w2	T	F	F

NOT(w2) = w1

Figure 4.5b. Omega-2 OR truth table.

Using the null substitution principle, and the interpretation of omega-2, the rule which governs the above truth tables can be simply stated.

> Principle 2: Any expression will evaluate to false for every possible substitution for an occurrence of omega-2.

Notice, also, in Figure 4.5b that the negation of omega-2 results in omega-1. The intuition behind this equivalence is that a value which is not inapplicable must be applicable. It can not be determined which applicable value is correct. This is exactly the definition of omega-1.

Operations Involving Omega-2

Figures 4.6b and 4.6c are examples of the operations of union and difference on tables containing

omega-2. The notion of functional dependency is retained in that the first column of each relation is assumed to be a prime attribute. in which no omega-2 is allowable.

P1

A	:	6
B	:	w2
C	:	3

P2

A	:	6
B	:	w2
D	:	w2
F	:	5

Figure 4.6a. Relations P1 and P2.

P1 U P2

A	6
B	w2
C	3
D	w2
F	5

Figure 4.6a. Union.

P2 - P1

D	w2
F	5

P1 - P2

C	3

Figure 4.6b. Difference.

Figures 4.7 and 4.8 are example relations for subsequent selections on tables containing omega-2. In the relation in Figure 4.8, the presence of omega-2 can be interpreted as an inapplicable attribute for that particular item. For example, part 101 comes in two colors, each with a different size. Part 102, however, is not or can not be described by color or size.

P3

Part_supl	Part#	Price
A	101	.03
A	102	.52
B	102	.50
B	103	.18
C	101	.04
C	102	.50
C	104	.25

Figure 4.7. Relation P3.

P4

Part#	Color	Size
101	white	10
101	red	22
102	w2	w2
103	silver	8
104	w2	22

Figure 4.8. Relation P4.

The following examples of selection illustrate that only those tuples which definitely match the selection criteria are returned. The truth tables defined for omega-2 are used to determine the results.

Select Color=white [P4]

Part#	Color	Size
101	white	10

Select Size=16 [P4]

Part#	Color	Size
-	-	-

Figure 4.9. Selection operation.

The first selection returns only one tuple, as only one value in the color attribute column matches the value "white". In the second selection, no tuples are returned because no values in the attribute column for size match the value "16".

Figure 4.10 shows a projection which illustrates a more important aspect of omega-2 processing.

P4 [Color, Size]

Color	Size
white	10
red	22
w2	w2
silver	8
w2	22

Figure 4.10. Projection operation.

The second and fifth tuples have the same value for size. Neither tuple can replace the other, however. They are not equivalent, by definition of omega-2.

The following example of the natural join is trivial (Figure 4.11). It does not show the behavior which is important in the processing of omega-2.

P3 |X| P4

Part_supl	Part#	Price	Color	Size
A	101	.03	white	10
A	101	.03	red	22
A	102	.52	w2	w2
B	102	.50	w2	w2
B	103	.18	silver	8
C	101	.04	white	10
C	101	.04	red	22
C	102	.50	w2	w2
C	104	.25	w2	22

Figure 4.11. Natural join of P3 and P4.

The following examples will give a better idea of the methods which bring about the desired results for the processing of omega-2. Figures 4.12 and 4.13 will be used to demonstrate the difficulties which may arise. Both columns of the join attribute have been left in place in the table in Figures 4.14 so that the processing concerns remain visible.

EMPLOYEES

Empname	Empnum	Phone
Smith	101	418
Jones	102	w2
Burns	103	327
Allen	104	w2
Laurel	105	611
Hardy	106	794
Jones	107	810
Moore	108	w2

Figure 4.12. Employee relation.

MANAGERS

Empname	Proj#	Phone
Jones	89	w2
Hardy	64	794
Burns	41	327

Figure 4.13. Manager relation.

EMPLOYEES |X| MANAGERS

Empname	Empnum	Phone	Empname	Proj#	Phone
Jones	102	w2	Jones	89	w2
Hardy	106	794	Hardy	64	794
Burns	103	327	Burns	41	327

Figure 4.14. Natural join.

All attributes that have the same names (domain), must match before a tuple may be formed for the natural join. In the case of the natural join above, "Empname" and "Phone" must match from each relation to become a tuple in the new table.

Theta joins are demonstrated in Figures 4.15a and 4.15b. Again, all attribute columns are left intact so that the join criteria may be examined.

EMPLOYEES [Phone = Phone] MANAGERS

Empname	Empnum	Phone	Empname	Proj#	Phone
Jones	102	w2	Jones	89	w2
Allen	104	w2	Jones	89	w2
Moore	108	w2	Jones	89	w2
Hardy	106	794	Hardy	64	794
Burns	103	327	Burns	41	327

Figure 4.15a. Theta join.

EMPLOYEES [Empname = Empname] MANAGERS

Empname	Empnum	Phone	Empname	Proj#	Phone
Jones	102	w2	Jones	89	w2
Jones	107	810	Jones	89	w2
Hardy	106	794	Hardy	64	794
Burns	103	327	Burns	41	327

Figure 4.15b. Theta join.

It is apparent that these results for the joins are in some manner incorrect. There is a need to determine the semantically correct method of removing those tuples which are obviously inconsistent. For example, in the first join, the names in tuples two and three do not match. Because of the identically named attributes involved in the relations, these must be taken into account although the join attribute does not involve the attributes. In the second join, the names match for all tuples, yet there is an inconsistency in the phone attribute. It is necessary to consider all attributes which have the same attribute name, otherwise, an inconsistency will result.

In Figure 4.15a, the second and third tuples should not be included in the results of the join. Likewise, in Figure 4.15b, the second tuple shows this

same inconsistency. It should be removed from the join before the results are returned.

The rules for duplicate removal and data extraction for omega-2 are the same. An exact match must occur in either type of processing.

Interaction of Omega-1 and Omega-2 Nulls

As omega-2 is interpreted as a domain value, distinct from all other domain values, when an omega-1 is compared against the value omega-2, it seems possible that omega-2 is a valid value for omega-1 to represent. However, by using the omega-1 as the value for that attribute, it is stated that some value does exist--excluding omega-2 as a possible value for the replacement of omega-1. This section will examine the interaction of these two nulls when present in the same relational table.

Part_supl	Part#
A	101
A	102
A	103
B	102
B	103
C	101
C	102

Figure 4.16. Parts-Supplier relation.

Part#	Color	Size
101	w2	w1
102	w1	12
102	blue	14
103	red	w1
103	blue	14

Figure 4.17. Part#-color-size relation.

The relational table in Figure 4.17 contains both omega-1 and omega-2 as values for attributes color and size. The figures below show a selection on this relation.

Select color = "red"

Part#	Color	Size
102	w1	12
103	red	w1

Figure 4.18a. Selection operation.

Select color /= "red"

Part#	Color	Size
101	w2	w1
102	w1	12
102	blue	14
103	blue	14

Figure 4.18b. Selection operation.

In the first selection, the presence of omega-1 as a value allows the result to include both tuples for which the color could possibly be "red". The second selection returns all tuples for which the color is not "red", and possibly not "red". This brings up an important point which adds to the usability of the concept being presented. The ability to represent negative information is very valuable. The non-truth functionality of Codd's original truth tables for the

omega null was first explained using such a negation-type query. As an example, the query "(Color not "red" AND Size = 12) OR (Color = "red" AND Part# = 103)" should return tuples one, two, and four, and, in fact, does when the truth tables defined for the respective nulls are used. Another problem mentioned was that of a ("red" and not "red)-type of query. The results of the both selections for these criteria can be seen to contain all the information available, which is the expected result.

Project [Color, Size]

Color	Size
w2	w1
w1	12
blue	14
red	w1

Figure 4.19a. Projection operation.

Two tuples containing <blue,14> are returned for this projection, but since they match exactly, the duplication is removed. Tuples one and two are definitely not possible duplicates as omega-2 is never equal to a possible domain value such as omega-1. It is also possi-

ble that tuple two and four represent the same information.

Project [Part#, Size]

Part#	Size
101	w1
102	12
102	14
103	w1
103	14

Figure 4.19b. Projection operation.

In the projection in Figure 4.19b, it is possible that tuples four and five are, in reality, the same information. Neither tuple is removed, however, to prevent hiding some possibly correct information.

Project [Part#, Color]

Part#	Color
101	w2
102	w1
102	blue
103	red
103	blue

Figure 4.19c. Projection operation.

The same concern of hiding possibly correct information is apparent in Figure 4.19c. Therefore, all tuples remain in the result as before.

[Part_supl, Part#] |X| [Part#, Color, Size]

Part_supl	Part#	Color	Size
A	101	w2	w1
A	102	w1	12
A	102	blue	14
A	103	red	w1
A	103	blue	14
B	102	w1	12
B	102	blue	w1
B	103	red	w1
B	103	blue	14
C	101	w2	w1
C	102	w1	12
C	102	blue	14

Figure 4.20. Natural join.

All possibly correct information has been returned by taking the natural join of the two relations in question. By refering to the original tables which were used to create this natural join, it can be seen that both tables are no longer needed to represent the information. Allowing omega-1's and omega-2's in this relation does away with the need for any other tables to represent the same information. As can be seen from the natural join relation above (Figure 4.20), more

information can be more precisely represented with the combined use of omega-1 and omega-2.

The Overall View of Omega-1 and Omega-2

Although there are many ways in which these nulls may be represented, the major objection to previous simplistic representations has been the non-truth functionality of the logical operations. This semantic representation of omega-1 and omega-2 allows much more information, particularly that which is only partially defined or inapplicable, to be included in processing in a useful manner.

Two types of null values representing unknown and inapplicable attribute values have been presented for inclusion in relational databases. Each null has been semantically defined with the use of logical operations which illustrate the truth functionality. The power of each representation has been demonstrated by examples of algebraic operations on a variety of relational tableaux. The omega-1 and omega-2 nulls have also been used in the same table to give a graphic example of the power of the suggested representation.

There are still many facets of this representation which can be futher developed. It is, as yet, only an intuitive approach to a problem which has been studied

in great detail. The strength of the approach lies in the simplicity of the overall concept and the consistency of the processing. The following chapter will include suggestions which may enhance the mechanisms developed here.

Chapter 5 : The Representation as a Usable Basis

The semantic definition given the omega-1 and omega-2 are modifications of the previously defined omega and theta nulls. Many attempts to formalize a usable system with nulls have been made, yet none return the results desired for this study. Those which have approached the issue from the standpoint of processing concerns rather than representational concerns, have developed very complex, impractical techniques. These techniques, while very formally developed, still do not display the desired representation or behavior. In addition, many are computationally very complex.

The mechanisms developed here exhibit the desired behavior in terms of processing and representation. All possibly correct information is presented as a result of an algebraic operation. The semantics of the nulls, in both cases, retain their consistency in the face of processing.

Truth Functionality

Truth functional processing has been presented through the development of the modified truth tables for omega-1 and omega-2. The examples shown in Chapter

four illustrate that the major concern of non-truth functionality has been removed with the representation of omega-1. The underlying meaning of omega-1 is preserved throughout various operations. Data extraction with omega-1's is accomplished with the use of the truth table defined for it. Duplicate removal follows the normal rule of removing only those tuples which are exact matches. Tuples resulting from an operation retain the semantics of the omega-1 null.

The view taken in this representation is that of a partially open world. Information is specified to the extent that a finite amount of information can be used in the model. Negative inferences can be taken from the presence of both types of null. The representation can be used to determine what information "is" and what information "isn't". This gives the advantage that the meaning of a null is considered on a wider basis than that of the individual tuple, as in some representations.

Tuples Resulting From Processing

Data extraction is the prime area where processed information might exhibit incorrect results from further processing. The rules developed for data extraction are quite simple and in line with the seman-

tio interpretation given the two nulls. By examining the results of algebraic operations, it is seen that the information is still intact. No information is lost and no extraneous information is introduced. This is extremely important when selection criteria become more complex.

Duplicate removal is the other aspect which might affect extended processing. It has not been proven, however that corruption of information will take place if the rule of exact matches for duplicate removal is used. No problems should result from the removal of a tuple which is exactly represented by another.

Further examination should be given both of these aspects of processing tables with omega-1 and omega-2. The intuitive appeal of the results shown point to proper behavior for extended processing.

Functional Dependency Considerations

One view of this representation which has not been thoroughly investigated is the constraints which are placed on information in the form of functional dependencies. As dependency theory is an intrinsic part of relational database theory, it will introduce constraints concerning the up-to-now unlimited appearance of nulls. Some of these considerations have already

been introduced, particularly in the discussion of allowable domains for omega-2.

Restricting the occurrence of nulls as specific attribute values is an important issue. To allow maximum flexibility of representation, no restriction has been placed on the omega-1 null, with minimal restriction placed on the omega-2 null. There is an advantage in allowing the representation of information such as, "some manufacturer makes item 102". It is understood that the identity is not known at this time. Of course, a retrieval on a specific manufacturer, say "A", would also result in the retrieval of this unknown manufacturer. This may not be desirable, but at the same time, it is not a semantic contradiction.

Integrity Constraints

A very brief example given in a later section of Chapter four illustrates the manner in which integrity constraints might be relaxed with the inclusion of these nulls. A database with a lossless schema may be used to demonstrate that less redundancy of information can be achieved in a representation relying upon this semantic definition of omega-1 and omega-2.

Complex Query Expressions

Some effort has been made to illustrate that evaluation of simple expressions results in the semantically correct retrieval of information. The fact that this basic set of algebraic operations works as expected suggests that few problems will develop as more complex expressions are applied. The operations illustrated in Chapter four can be combined. Although it has not been proven, it is felt that compound expressions should not introduce incorrect information.

No investigation has been done concerning optimization of queries. Algorithms for both aspects of processing, data extraction and duplicate removal, should be simple and efficient to implement.

Statistical Representations

Perhaps one of the most interesting aspects of this representation is its inclusion in some type of decision support system. This type of system produces estimates from partially known information. The incorporation of an associated probability for a tuple containing some partially known information is a future consideration. For example, if it is known that the probability of a certain fact being correct is "x", an estimate could be established for the use of this

information in combination with other factors.

Conclusions

This representation of two distinct null values exhibits many of the behavioral qualities which are of importance in the modeling of an incompletely known world for information processing. The logical and algebraic operators which have been defined are consistent with the semantics of the representation. The intuitive aspect of this study does not negate its usability and flexibility for future investigation.

Appendix a.

Manifestation of Nulls

-Not valid for this individual (e.g., maiden name of male employee)

-Valid, but does not yet exist for this individual (e.g., married name of female unmarried employee)

-Exists, but not permitted to be logically stored (e.g., religion of this employee)

-Exists, but not knowable for this individual (e.g., last efficiency rating of an employee who worked for another company)

-Exists, but not yet logically stored for this individual (e.g., medical history of newly hired employee)

-Logically stored, but subsequently locially deleted

-Logically stored, but not yet available

-Available, but undergoing change (may be no longer valid)

- Change begun, but new values not yet computed
- Change incomplete, committed values are part new, part old, may be inconsistent
- Change incomplete, but part new values not yet committed
- Change complete, but new values not yet committed

-Available, but of suspect validity (unreliable)

- Possible failure in conceptual data acquisition
- Possible failure in internal data maintenance

-Available, but invalid

- Not too bad
- Too bad

-Secured for this class of conceptual data

-Secured for this individual object

-Secured at this time

-Derived from null conceptual data (any of the above)

BIBLIOGRAPHY

[ANS75] ANSI/X3/SPARC Study Group on Data Base Management Systems, Interim Report, ANSI, February, 1975.

[Bis81] Biskup, Joachim : "A Formal Approach to Null Values in Database Relations", In Advances in Database Theory, Vol. 1, Gallaire, Plenum, 1981, pp. 299-341.

[Bis83] Biskup, Joachim : "A Foundation of Codd's Relational Maybe-Operations", ACM Transactions on Database Systems, Vol. 8, No. 4, Dec. 1983, pp. 608-636.

[Bow84] Bowers, David S. : "A Database Architecture for Aggregate-Incomplete Data", The Computer Journal of the British Computer Society, Vol. 27, No. 4, November 1984, pp. 294-300.

[Buc82] Buckles, Billy P., and Frederick E. Petry : "A Fuzzy Representation of Data for Relational Databases", Fuzzy Sets and Systems 7, 1982, pp. 213-226.

[Bue82] Buell, Duncan A. : "An Analysis of Some Fuzzy Subset Applications to Information Retrieval Systems", Fuzzy Sets and Systems 7, 1982, pp. 35-42.

[Cod79] Codd, E.F. : "Extending the Database Relational Model to Capture More Meaning", ACM Transactions on Database Systems, Vol. 4, No. 4, Dec. 1979, pp. 397-434.

[Cod75] Codd, E.F. : "Understanding Relations", FDT (ACM SIGMOD Records), Vol. 7, No. 3-4, 1975, pp. 23-28.

[Col75] Collins, Allan, E. Warnock, N. Aiello, M. Miller : "Reasoning from Incomplete

Knowledge", in Representation and Understanding, Studies in Cognitive Science, Bobrow and Collins, Ed., 1975, Academic Press, Inc., pp. 383-414.

[Col81] Colmerauer, A, and J.F. Pique : "About Natural Logic", In Advances in Database Theory, Vol. 1, Gallaire, Plenum, 1981, pp. 343-365.

[Dav80] Davis, Martin : "The Mathematics of Non-Monotonic Reasoning", Artificial Intelligence 13, 1980, pp. 73-80.

[Fag82] Fagin, Ronald, A.O. Mendelzon, and J.D. Ullman : "A Simplified Universal Relation Assumption and Its Properties", ACM Transactions on Database Systems, Vol. 7, No. 3, Sept. 1982, pp. 343-360.

[Gai81] Gaines, Brian R. : "Logical Foundations for Database Systems", in Fuzzy Reasoning and its Applications, (Mamdani and Gaines, Ed.), Academic Press, London, 1981, pp. 289-308.

[Gol81] Goldstein, Billie S. : "Constraints on Null Values in Relational Databases", VLDB, 1981, pp. 101-110.

[Gra77] Grant, John : "Null Values in a Relational Database", Info. Proc. Letters, Vol. 6, No. 5, 1977, pp. 156-157.

[Gra79] Grant, John : "Partial Values in a Tabular Database Model", Info. Processing Letters, Vol. 9, No. 2, Aug. 17, 1979, pp. 97-99.

[Gra80] Grant, John : "Incomplete Information in a Relational Database", Fundamenta Informaticae, Vol.3, 1980, pp. 363-378.

[Imi84] Imielinski, Tomasz, and Witold Lipski, Jr. : "Incomplete Information in Relational Databases", JACM, Vol. 31, No. 4, Oct. 1984, pp. 761-791.

[Imi81] Imielinski, Tomasz, and Witold Lipski, Jr. : "On Representing Incomplete Information in a Relational Database", VLDB, 1981, pp. 388-397.

[Imi83] Imielinski, Tomasz, and Witold Lipski, Jr. : "Incomplete Information and Dependencies in Relational Databases", ACM SIGMOD International Conference on Management of Data, San Jose, 1983, pp. 178-184.

[Jae78] Jaegermann, M. : "Information Storage and Retrieval Systems with Incomplete Information I, Fundamenta Informaticae II, (1978), 17-41.

[Jae79] Jaegermann, M. : "Information Storage and Retrieval Systems with Incomplete Information II, Fundamenta Informaticae II, (1979), 141-166.

[Jae82] Jaeschke, G. : "The Theory of One Attribute Nesting", Heidelberg Scientific Center Technical Note, TN 82.01.

[Lev81] Levesque, Hector J. : "The Interaction with Incomplete Knowledge Bases: A Formal Treatment", Proc. of the 7th International Joint Conference On Artificial Intelligence, Vol. 1, August 1981, pp. 240-245.

[Lie79] Lien, Y. Edmund : "Multivalued Dependencies with Null Values in Relational Databases", VLDB, 1979, pp. 61-66.

[Lip76] Lipski, Witold, Jr. : "Informational Systems with Incomplete Information", Proceedings of the 3rd International Colloquium on Automata, Languages and Programming (Edinburgh, Scotland, July 20-23), Edinburgh University Press, Edinburgh Scotland, 1976, pp. 120-130.

[Lip81] Lipski, Witold, Jr. : "On Databases with Incomplete Information", JACM, Vol.28, No. 1, Jan. 1981, pp. 41-70.

[Lip79] Lipski, Witold, Jr. : "On Semantic Issues Connected with Incomplete Information Databases", ACM Transactions on Database Systems, Vol. 4, No. 3, September 1979, pp. 262-296.

[McC80,1]McCarthy, John : "Circumscription--A Form of Non- Monotonic Reasoning", Artificial Intelligence 13, 1980, pp. 27- 39.

[McC80,2]McCarthy, John : "Addendum: Circumscription and Other Non-Monotonic Formalisms", Artificial Intelligence 13, 1980, pp. 171-172.

[McD80] McDermott, Drew, and Jon Doyle : "Non-Monotonic Logic I", Artificial Intelligence 13, 1980, pp. 41-72.

[Osb81] Osborn, Sylvia : "Insertions in a Multi-Relation Database with Nulls", IEEE COMPSAC, 1981, pp. 75-80.

[Rei78] Reiter, Raymond : "On Reasoning by Default", Theoretical Issues in Natural Languages Processing-2, July 1978, pp. 210-218.

[Rei84] Reiter, Raymond : "Towards a Logical Reconstruction of Relational Database Theory", in Conceptual Modelling, Perspectives from Artificial Intelligence, Databases and Programming Languages, M.L. Brodie, J. Mylopoulos, and J. Schmidt, Eds., Springer-Verlag, New York, 1984, pp. 191- 233.

[Rei81] Reiter, Raymond, and Giovanni Criscuolo : "On Interacting Defaults", Proc. of the 7th International Joint Conference on Artificial Intelligence, Vol. 1, August 1981, pp. 270-276.

[Ris77] Rissanen, Jorma : "Independent Components of Relations", ACM Transactions on Database Systems, Vol. 2, No. 4, December 1977, pp. 317-325.

[Sag81] Sagiv, Yehoshua : "Can We Use The Universal

Instance Assumption Without Using Nulls?", ACM SIGMOD, 1981, pp. 108-120.

[Sci80] Sciore, Edward : "The Universal Instance and Database Design", Doctoral Thesis, 1980.

[Sik81] Siklossy, L. : "Efficient Query Evaluation in Relational Databases with Missing Values", Info. Processing Letters, Vol. 13, No. 4,5, End 1981, pp. 160-162.

[Vas79] Vassiliou, Yannis : "Null Values in Database Management-- A Denotational Semantics Approach", Proceedings SIGMOD Conference, May-June 1979, pp. 162-169.

[Vas80] Vassiliou, Yannis : "Functional Dependencies and Incomplete Information", VLDB, 1980, pp. 260-269.

[Win80] Winograd, Terry : "Extended Inference Modes in Reasoning by Computer Systems", Artificial Intelligence 13, 1980, pp. 5- 26.

[Won82] Wong, Eugene : "A Statistical Approach to Incomplete Information in Database Systems", ACM Transactions on Database Systems, Vol. 7, No. 3, Sept. 1982, pp. 470-488.

[Zad83] Zadeh, L.A. : "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems", Fuzzy Sets and Systems 11, 1983, pp. 199-227.

[Zan82] Zaniolo, Carlo : "Database Relations with Null Values (Extended abstract)", Proceedings of the ACM Symposium on Principles of Database Systems, March, 1982, pp. 27-33.

THE USE OF NULL VALUES IN A RELATIONAL DATABASE TO REPRESENT INCOMPLETE AND INAPPLICABLE INFORMATION

by

MARIA MARSHALL WILSON

B. Ed., Washburn University, 1973

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1985

An increasing concern in relational database theory is that of retaining representational consistency while allowing additional semantic representation. A usable representation of null values is important in modeling possible real-world situations where unavailable information can prevent the inclusion of any associated information in processing. Current research has developed methods which involve complex processing techniques for evaluation of queries, which eliminate the benefits of the ability to use null values in the representation. Other research has changed the representation mechanism to allow the inclusion of nulls in relational tables.

A new interpretation and a simple processing mechanism is defined for the inclusion of nulls. Two specific null values are defined: the null representing incomplete knowledge, "omega- 1", and the null representing inapplicable information, "omega-2". These interpretations will be of use in the development of a relational database which will accept partial information for storage, and incorporate this incomplete data in the evaluation of queries. A set of

relational operations is developed which allows the inclusion and processing of null values, while preserving the integrity of the underlying information.